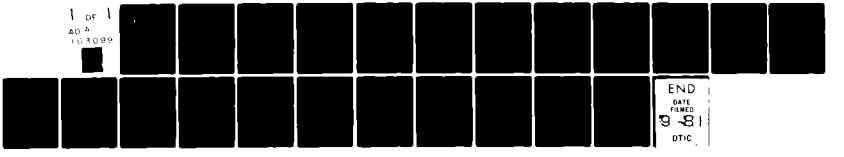
AD-A103 099 KANSAS STATE UNIV MANHATTAN DEPT OF COMPUTER SCIENCE F/G 9/2
RESEARCH IN FUNCTIONALLY DISTRIBUTED COMPUTER SYSTEMS DEVELOPME--ETC(U)
APR 76 P S FISHER, F J MARYANSKI DAAG29-76-G-0108
UNCLASSIFIED CS-76-11 NL



END
DATE FILMED
9-81
DTIC

AIRMICS

Army Institute for Research in Management Information and Computer Science

313 Calculator Bldg.
GA Institute of Technology
Atlanta, GA 30332

Technical Report

RESEARCH IN FUNCTIONALLY DISTRIBUTED COMPUTER SYSTEMS DEVELOPMENT

Kansas State University

Virgil Wallentine

Principal Investigator

VOLUME XVI

A MINICOMPUTER BASED DISTRIBUTED DATA BASE SYSTEM

U.S. ARMY COMPUTER SYSTEMS COMMAND FT BELVOIR, VA 22060

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A103099
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) A MINICOMPUTER BASED DISTRIBUTED DATA BASE SYSTEM.	5. TYPE OF REPORT & PERIOD COVERED Interim
	6. PERFORMING ORG. REPORT NUMBER CS-76-11
7. AUTHOR(s) Paul S. Fisher, Fred Maryanski, Virgil E. Wallentine, Myron A. Calhoun, Louis Seinsuitz	8. CONTRACT OR GRANT NUMBER(s) DAAG 29-76-G-0108
9. PERFORMING ORGANIZATION NAME AND ADDRESS Kansas State University, Department of Computer Science, Manhattan, KS 66506	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Research Office, P O Box 12211, Research Triangle Park, NC 27700	12. REPORT DATE Apr 76
	13. NUMBER OF PAGES 20 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) US Army Computer Systems Command, Attn: CSCS-AT, Ft. Belvoir, VA 22060	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

DBMS
Minicomputers
DDBMS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

-over-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

-ABSTRACT-

This paper described a data base management system under development at Kansas State University, intended for use in a network composed primarily of minicomputers. The report presents a description of the computers forming the network and their intercomputer communication system. The data base management system is a network type as specified by CODASYL. An extension of a CODASYL-type DBMS to multicomputer configurations is presented and several DBMS network topologies are discussed. We then conclude with a discussion of a completely distributed data base network.

A Minicomputer Based Distributed Data Base System

April 1976

Technical Report #CS-76-11

Fred J. Maryanski
Paul S. Fisher
Virgil E. Wallentine
Myron A. Calhoun
Computer Science Department
Kansas State University
Louis Sernovitz
United States Army Computer Systems Command
Ft. Belvoir, VA

[1]The work report herein is supported by the U.S. Army Research Office Grant No. DAAD-29-76-G-0108

Abstract

This paper describes a data base management system under development at Kansas State University, intended for use in a network composed primarily of minicomputers. The report presents a description of the computers forming the network and their inter-computer communication system. The data base management system is a network type as specified by CODASYL. An extension of a CODASYL-type DBMS to multicomputer configurations is presented and several DBMS network topologies are discussed. We then conclude with a discussion of a completely distributed data base network.

I. Introduction

Many organizations have the need to expand their data processing power and simultaneously increase their data accessibility. This paper presents the results of research aimed at developing systems to satisfy that need. A distributed data base management system residing on a network composed mainly of minicomputers is under development by Kansas State University and the U. S. Army Computer Systems Command. A distributed data base system provides the capability for a user program to be entered on any machine in the system, executed by another (or the same) processor, and access data on all secondary storage devices attached to the network (provided security requirements are met).

A distributed data base system of the form presented in this report provides an economical and easily expandable data processing facility with considerable flexibility and computational power.

II. Data Base Network

A. Hardware Configuration

The network which supports the distributed data base system is composed of six computers manufactured by four different vendors. This heterogenous blend of architectures requires that the data base and communication software be portable. The software systems presented in this paper are designed to be sufficiently general to allow a wide variety of different computers to be incorporated into the network.

The present network configuration is depicted in Figure 1. The three INTERDATA machines and the NOVA reside in the Computer Science Department of Kansas State University. These four local machines are all directly connected via high speed links. The IBM 370/158 is located at the KSU Computing Center and is accessed via a slow speed phone link. The PDP 11/70 is situated at

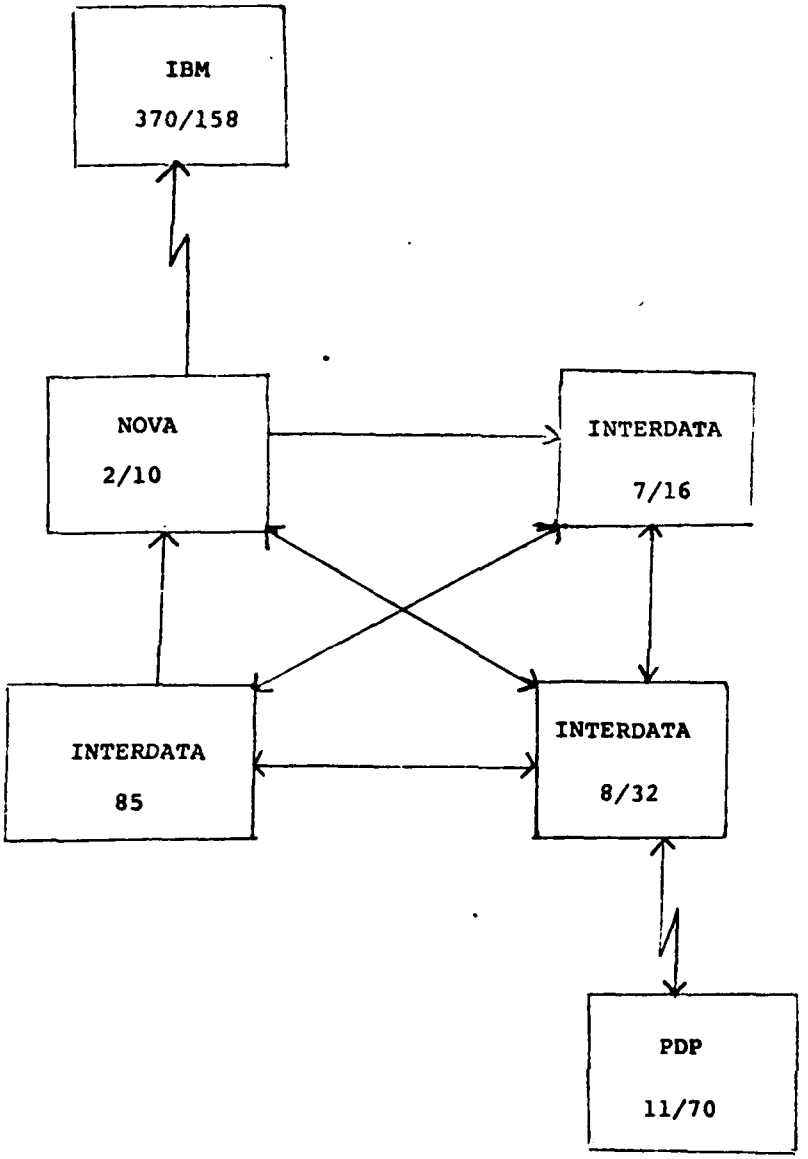


Figure 1

Network Configuration

Ft. Belvoir, VA (U. S. Army Computer System Command). The PDP 11/70 and INTERDATA 8/32 communicate over conventional phone lines.

B. Inter-Machine Communication

Although the network consists of a heterogeneous blend of processors, the means of communicating among the nodes is identical at the highest level in all cases. The Inter-Computer Communication System (ICCS) acts as a message system to route programs, data, and control information among machines and tasks. A functional overview of ICCS is given in this report. A detailed description can be obtained in reference [1]. ICCS performs the following functions:

1. Synchronize task and processor,

2. Perform inter-process communication which enables exchange of data and control information between distributed tasks,

3. Manage message buffers,

4. Standardize the interface between application programs and service programs (DBMS tasks) and,

5. Provide a uniform communication facility abstracted above but implemented on standard connections.

Inter-task communications is performed by SEND and RECEIVE procedures. The SEND procedure identifies the name and port within the target task addressed by a TO-ID parameter. The RECEIVE procedure may either receive a message from a task identified by a FROM-ID parameter or accept a message using a first-come-first-serve discipline. The RECEIVE procedure can also be notified if the task issuing the receive is to wait for the message or is to proceed unconditionally.

The interaction of ICCS, communicating tasks on different machines, and the operating systems of the machines is depicted in Figure 2. The task originating the communication invokes ICCS with a sequence of fixed length

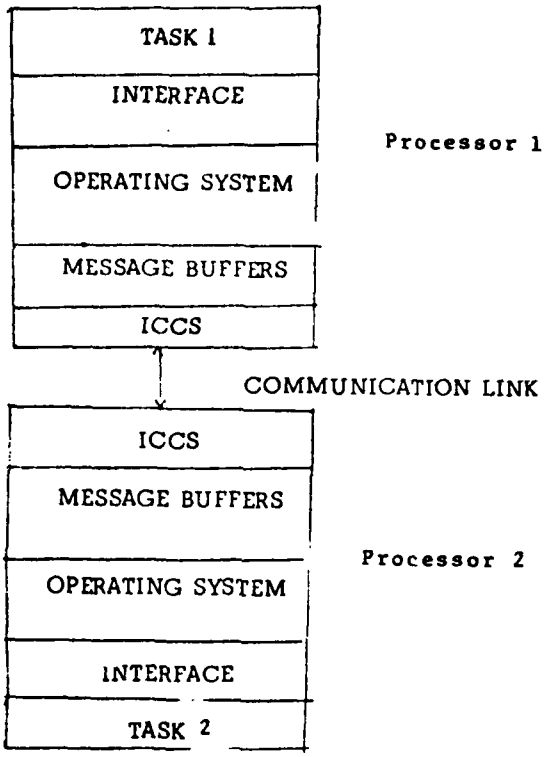


Figure 2
Inter-Task Communication via ICCS

block messages which compose the data or control information to be transmitted. The invocation is carried out be means of a subroutine call. The version of ICCS residing on the transmitting processor either transmits the series of buffers to the receiving machine via the communication links, or queues the messages if it has insufficient buffer space available. The message is then transmitted to the receiving processor which is specified by the message identifier. The transmission may be accomplished via direct links or by routing through other processors. The message system on the receiving processor reads the message into any available buffers. If no buffers are available the message is queued. The contents of the buffer are then made available to the task to which the message is directed.

When a task is to receive a message it issues a call to ICCS. If the message is not contained in the ICCS buffers in that processor, the task can specify either the wait or proceed option. If the wait option is indicated, the task will suspend execution until the ICCS on its processor receives the message. At this time the receiving task will be allowed to resume.

ICCS may exist in one of two forms depending upon the level of system software that can be utilized. The Multi-Computer Communication System (MCCS) is a version of the message control system intended to execute under single task operating systems. The Inter-Task Communication System (ITCS) is a multi-tasking version which executes on a system providing efficient skeletal inter-task communication facilities. MCCS has been implemented on the IBM 370 as a CMS machine under VM/370. Implementation details can be found in [2]. ITCS is in execution on a NOVA 2/10 under RDOS. Specifications of ITCS are given in [3].

ICCS has been developed to provide a generalized communication mechanism for intertask communication between distributed tasks. It can be adapted easily to serve as the communication facility for a data base management system. In

a DBMS environment, data base function requests can be transmitted in one direction and data and status sent in the other direction.

C. DBMS Specifications

The data base management system that is to be distributed functionally among the network nodes is based upon the CODASYL data base specifications. The DBMS encompasses virtually all of the features listed by the CODASYL committee as shown in Reference [4]. Complete language specifications for this system are available in Reference [5].

The internal operation of a CODASYL DBMS is illustrated here by describing the actions that occur when a DML statement is executed. More information on the workings of a CODASYL DBMS can be found in [6-8]. Figure 3 shows the memory layout of the DBMS and the action sequence.

1. A DML command is encountered in the application program.
   A call to the DBMS is then issued.
2. The DBMS analyzes the call and verifies the request against the object versions of the schema and sub-schema.
3. The contents of System Buffers are checked.
4. If necessary, the DBMS requests that the operating system perform a physical I/O transfer.
5. The operating system controls the I/O operation of 6.
6. Data is transferred between secondary storage and system buffers by the operating system.
7. The DBMS transfers data between system buffers and the User Working Area (UWA) as required.
8. The DBMS provides status information on the recently completed operation.
9. The data in the UWA may be operated upon in any manner by the application program.

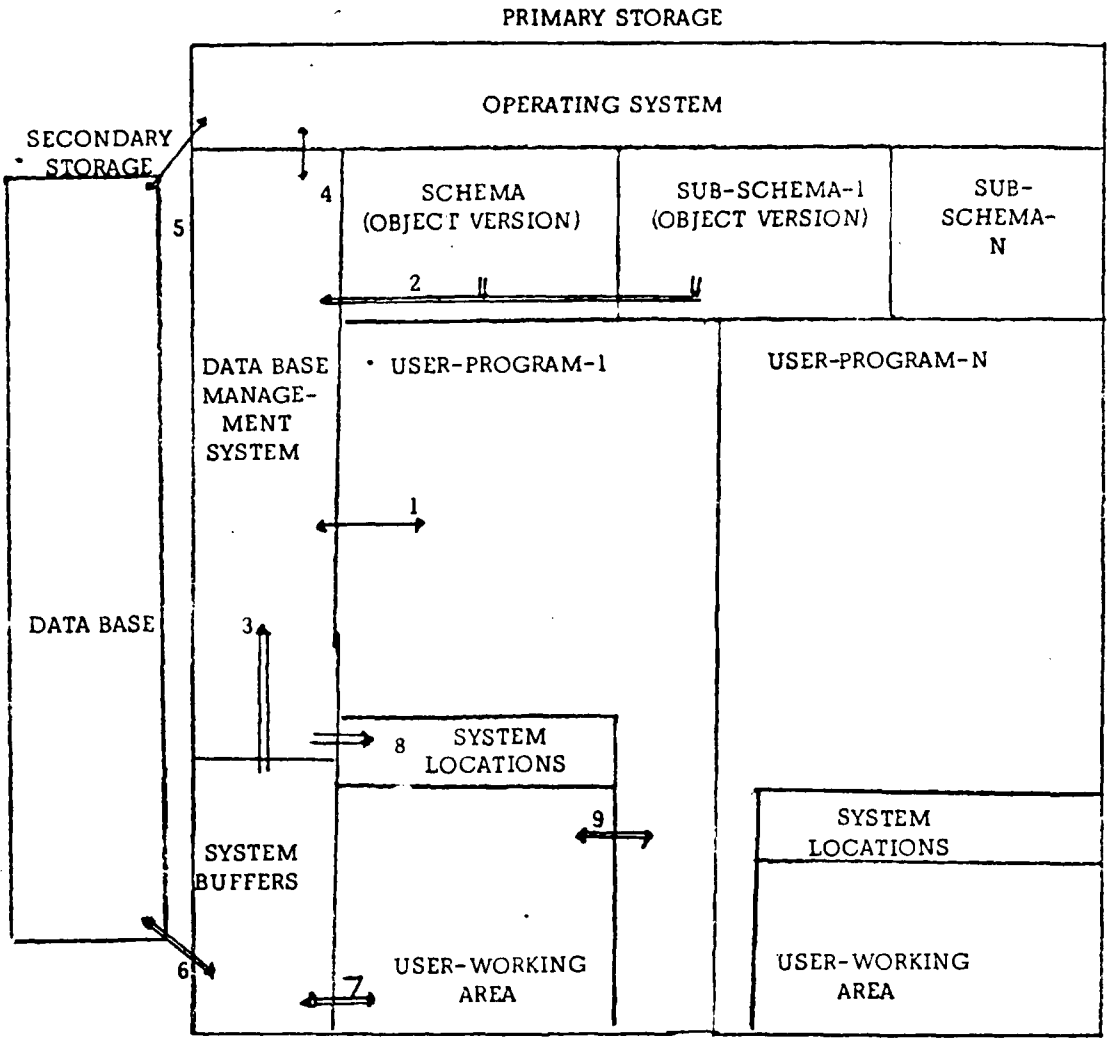


Figure 3
DBMS Memory Layout
and Operation

Control flow

Data flow

III. Distributed Data Base Systems

A. Back-End DBMS

A CODASYL DBMS as originally conceived was targeted for a single computer. However, a DBMS represents a significant drain on system resources due to the large amounts of computational activity and the sizeable number of I/O operations coupled with the operating system overhead introduced by the requisite task switches. Because of the high-level nature of the DML, a single DML statement can result in several secondary storage accesses, and hence many task switches are inherent in a centralized DBMS. A back-end DBMS was conceived by Canaday, et al. [9] to relieve the processor of a portion of its DBMS workload and overhead by incorporating a minicomputer dedicated to performing DBMS functions into the configuration.

The basic method of operation of a back-end DBMS is to restrict the physical access to the data base to the back-end machine. The application program is executed on the original (or host) computer. When a DML statement is encountered in an application program, a message is transmitted to the back-end computer via a communication system such as the ICCS described in Section II.B. A task on the back-end computer then performs the DML function, including any requisite I/O operations on the data base. In effect, the back-end processor acts as a sophisticated data base I/O device for the host machine.

The applicability of the back-end DBMS concept to production data processing systems has been investigated in [10]. In that study it was shown that the incorporation of a back-end machine into a DBMS system reduces task switching overhead on the host CPU, decreases the primary memory requirements for the DBMS and applications in the host, and provides for an overall increase in the availability of system resources.

In order to realize a back-end DBMS, the data base software shown in Figure 2 must be distributed between the host and back-end computers. The back-end configuration is shown in Figure 4. Each DML results in exchange of information between the two machines. The DBMS software in the host becomes minimal, consisting only of interface routines between the application program and the message system.

The back-end DBMS unloads substantial amount of processing requirement from the host to the back-end machine. This fact frees the host machine for additional processing. If this processing is dedicated to additional DBMS applications, the benefits of concurrent operation of the host and back-end CPU's can be repeated.

Inter-machine communication is accomplished in basically the following way. Whenever a DML statement results in a transmission to the back-end, an interface routine formats a message which indicates the action that the back-end must take in order to complete the DML function. The message is transmitted by the ICCS to the back-end. The back-end computer also contains a routine that serves as an intermediary between ICCS and the DBMS routines residing on the back-end. This routine will decode the message and activate the appropriate DBMS function which may result in one or more I/O operations. When the DML function has been completed, data and/or status information is returned to the application program in the host machine. Figure 5 indicates the flow of messages in the back-end DBMS.

B. Multiple Machine Configurations

Thus far, the discussion has considered only a two-computer data base network. However this configuration can be extended in several ways. An initial step in this direction is to incorporate several back-end machines into the system. In this environment the host serves as the central element of the network. Any data base access request originated by the host computer

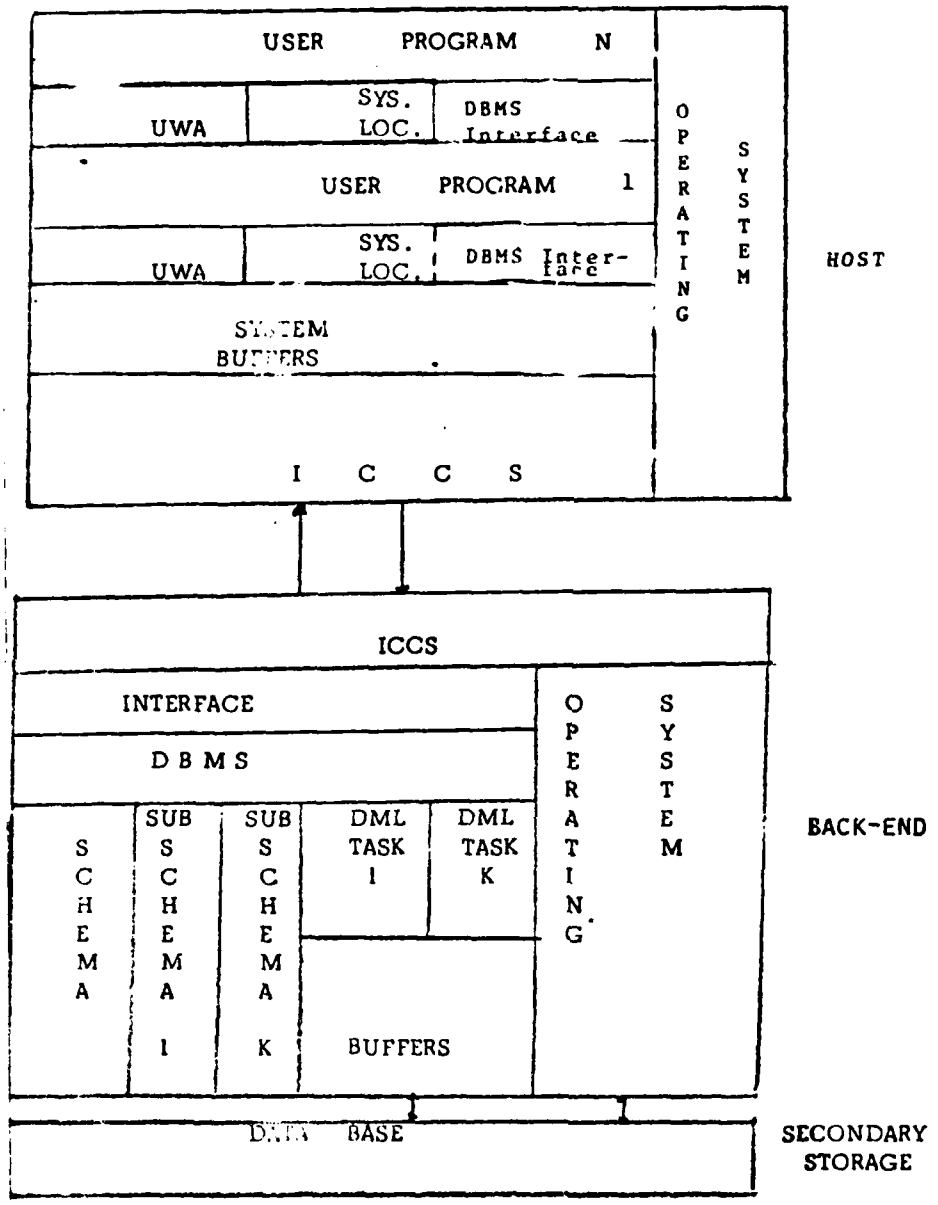


Figure 4

Back-end Software Distribution

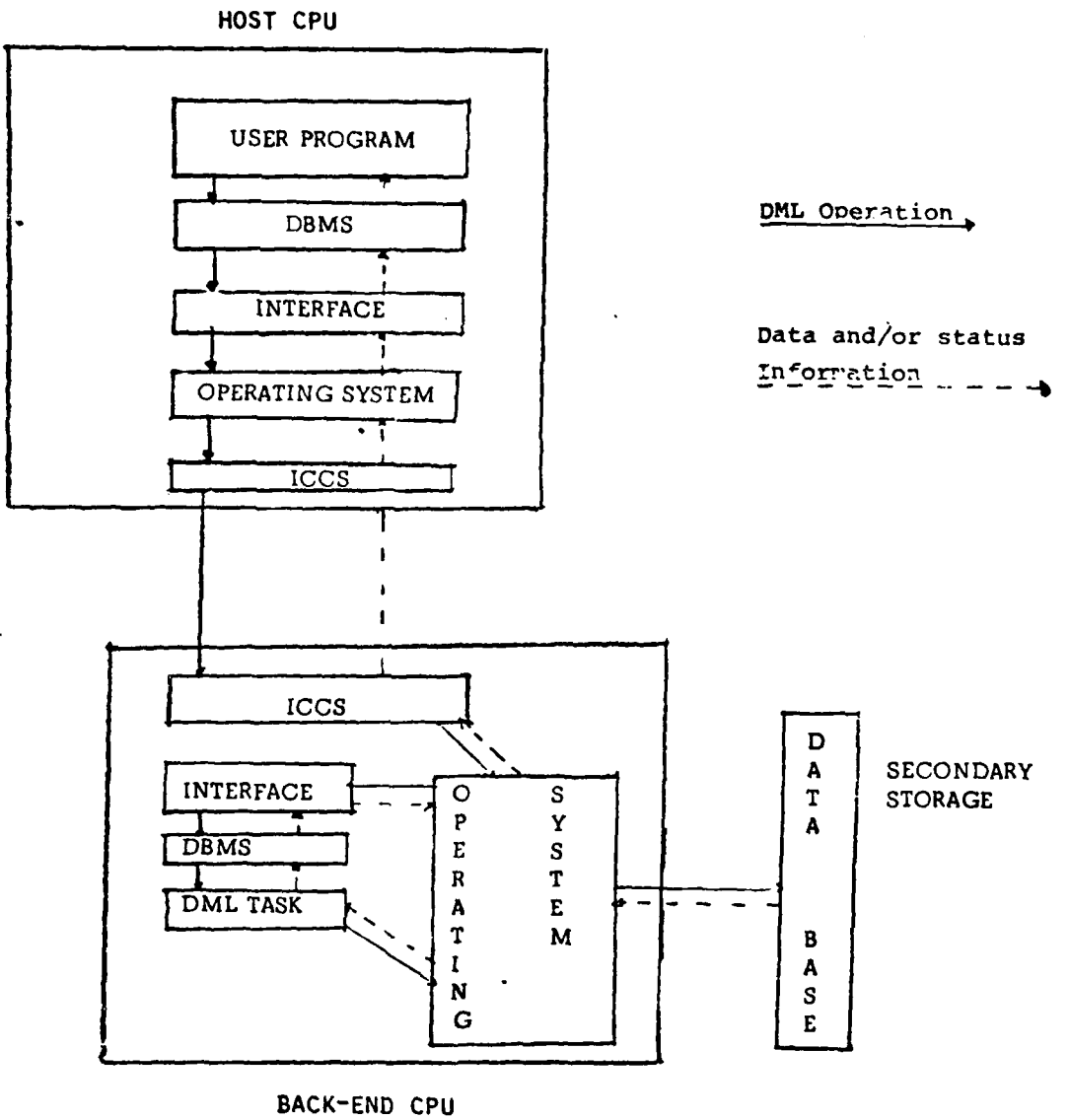


Figure 5
Information Flow in Back-End DBMS

must be targeted to the back-end machine connected to the appropriate data base. As in all network DBMS topologies, intermachine communication occurs via messages transmitted using the ICCS. Figure 6 illustrates a multiple back-end configuration.

The basic back-end system can also be expanded to allow several host machines to access a single back-end. In this environment, the back-end machine controls the access to a centralized data base which may be referenced for any of the host machines. Figure 7 depicts a multiple host configuration.

All prior discussions of the back-end machine have assumed that its sole function is data base management. One of the basic objectives of a distributed DBMS is efficient utilization of all system resources. It is possible that a CPU totally dedicated to the DBMS function could have considerable periods of inactivity. Assuming that the back-end computer has multiprogramming capabilities, tasks other than DBMS functions could be performed upon that machine. These additional tasks could include data base application programs. A machine capable of serving as both a host and back-end is known as a bi-functional machine. In a network with back-end, host and bi-functional machines (such as that in Figure 8) the only restriction as to the function of a processor is its physical connection to secondary storage. A bi-functional processor does not require any special software other than the DBMS and the ICCS code. If an application program on a bi-functional machine has access to the data base controlled by that machine, messages are transmitted to and from the back-end DBMS code on the same machine via the ICCS of that machine. This mode of operation naturally introduces overhead with respect to a single processor DBMS. However, the benefits to be realized in terms of generality of function and expanded data access make such a configuration highly desirable.

The goal of a distributed computing system is to balance the workload of the component computing elements while maximizing access and throughput. In a

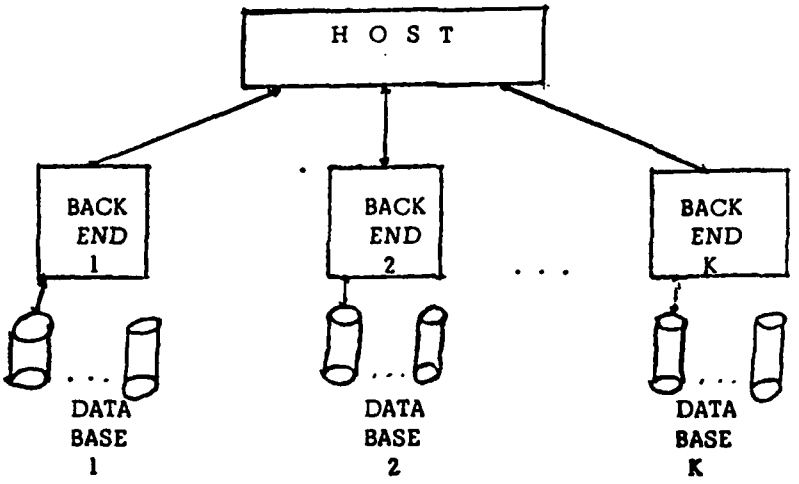


Figure 6
Multiple Back-End Configuration

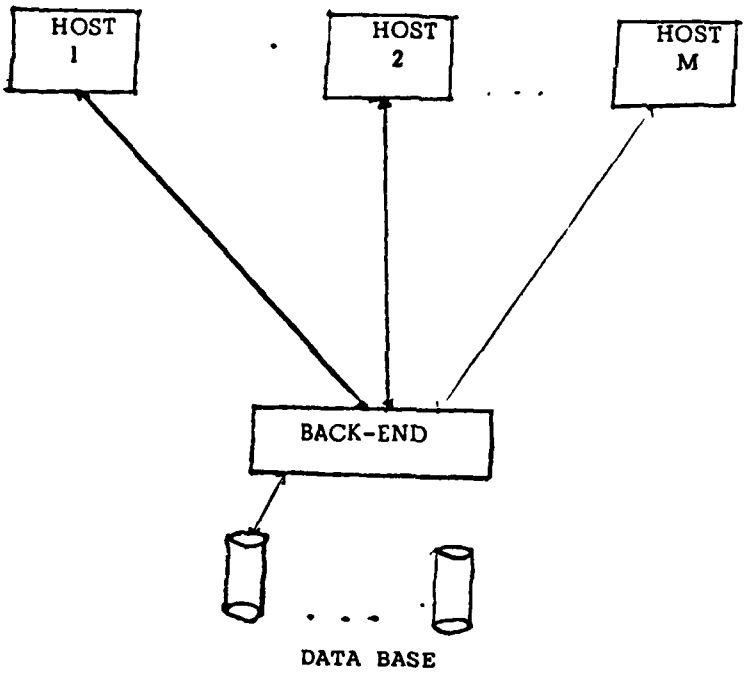


Figure 7.
Multiple Host Configuration

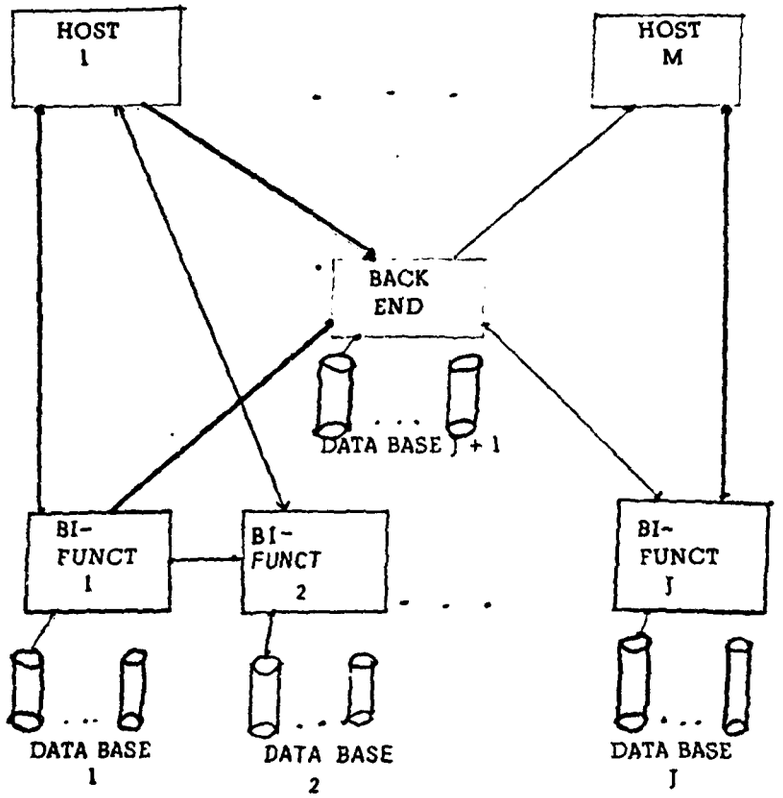


Figure 8
DMBS Network with Bi-Functional Machines

distributed DBMS network, each node is a bi-functional machine. Each node may communicate with every node in the system. (Although this communication may be realized by forwarding through intervening nodes.) Figure 9 displays a typical distributed DBMS. In such a configuration, an application program may be submitted at one node, executed at another, and access the data bases of other nodes in the network.

Problems such as data accessibility and integrity, transparency of data location, contention and deadlock, communication and buffer management for a distributed DBMS are considered in the references [11-12].

IV. Conclusion

In summary this paper provides a description of a flexible and powerful network for data base management. The report describes the mini-computer configuration upon which the system resides, the inter-machine communication facilities, a description of the operation of the data base management system and discussion of various network configurations under which the DBMS can operate.

This report presents a distributed data base network as an economical, general, and practical method of enhancing (or expanding) a data processing facility.

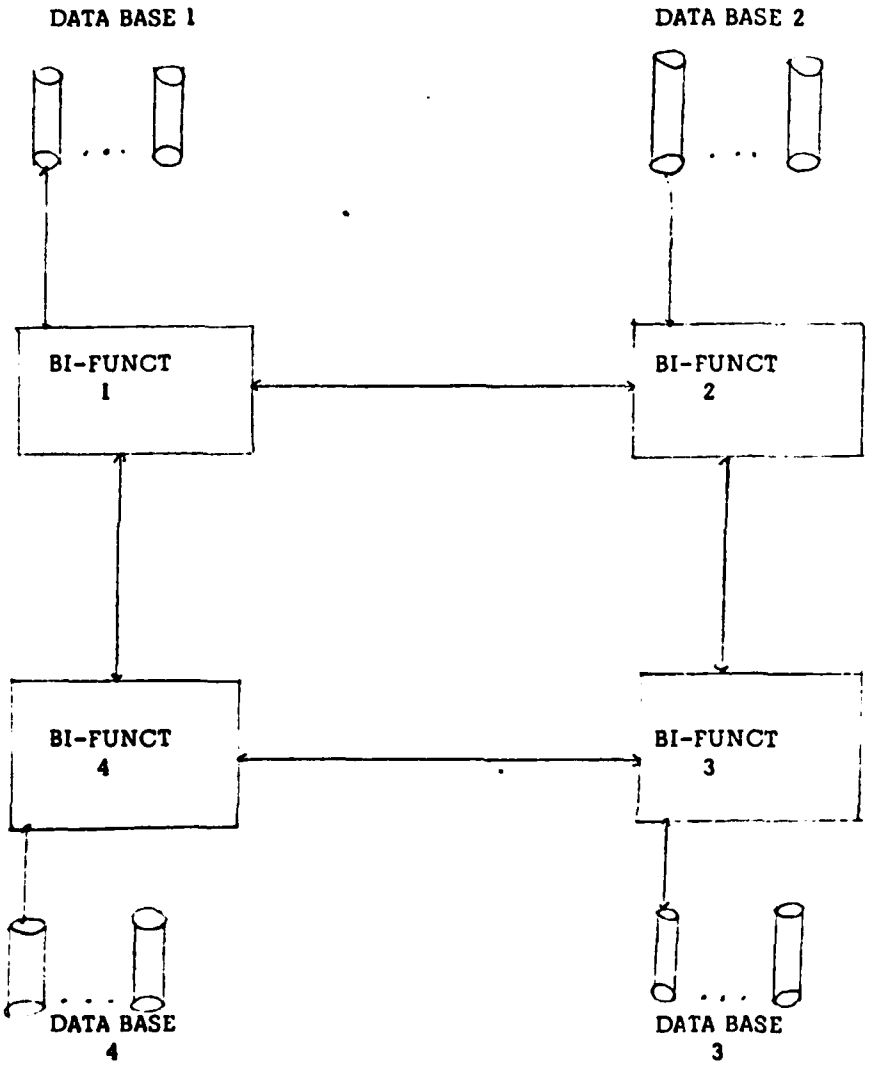


Figure 9

Distributed DBMS

References

1. Wallentine, V.E. and Maryanski, F.J., Implementation of a Distributed Data Base System, TR-CS-11-75, Computer Science Dept., Kansas State University, Manhattan, Kansas 66506, Nov., 1975.

2. Fox, S., A Multi-Computer Communication System, M.S. Report, Computer Science Dept., Kansas State University, Manhattan, Kansas 66506, Jan. 1976.

3. Wallentine, V.E., et al., On the Implementation of a Back-end Data Base Management System, TR-CS-09-75, Computer Science Dept., Kansas State University, Manhattan, Kansas 66506, Sept., 1975.

4. CODASYL COBOL 1973 Journal of Development, Dept. of Supply and Services, Technical Service Branch, Ottawa, Ontario (revisions to June, 1975).

5. Maryanski, F.J. and Fisher, P.S., Language Specifications for a Distributed Data Base Management System, Technical Report, Computer Science Dept., Kansas State University, May, 1976.

6. Date, C.J., Introduction to Data Base Management, Addison Wesley, Reading, Mass., 1975.

7. Martin, J., Computer Data Base Organization, Prentice-Hall, Englewood Cliffs, N.J., 1975.

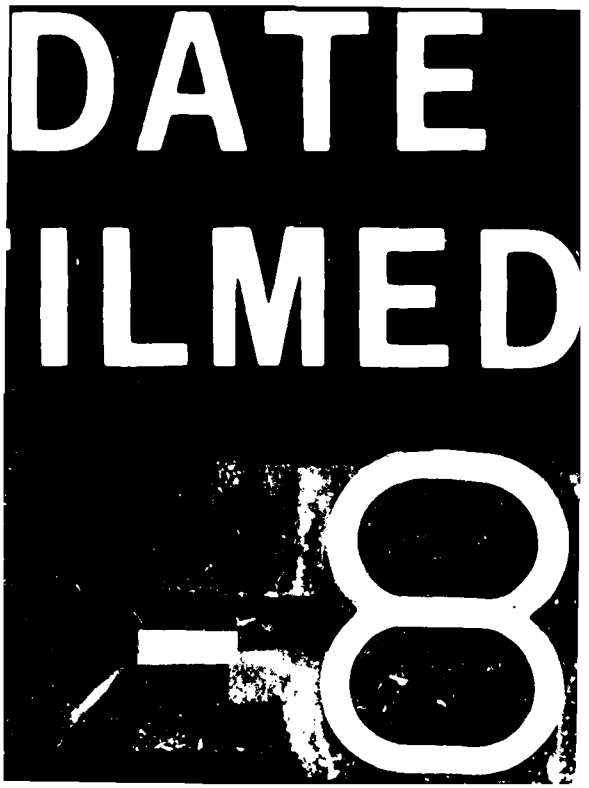
8. Warren, T., Feature Analysis of CODASYL Data Base Management Systems, AD-A014 972, NTIS, Dept. of Commerce, Springfield, VA, June, 1975.

9. Canaday, R.H., et al., A Back-end Computer for Data Base Management, CACM 17, 10, Oct., 1974, pp. 575-582.

10. Maryanski, F., Fisher, P., and Wallentine, V., Feasibility of Back-End Minicomputers, USACSC Grant No. DAHC04-75-6-0137, U.S. Army Computer Systems Command, Ft. Belvoir, VA 22060, June, 1975.

11. Maryanski, F.J., Fisher, P.S. and Wallentine, V.E., Distributed Data Base Management Systems, Technical Report, Computer Science Dept., Kansas State University, Manhattan, KS 66506 (in prep.).

12. Maryanski, F.J. and Wallentine, V.E., Resource Sharing in a Distributed Data Base Management System, Technical Report, Computer Science Dept., Kansas State University, Manhattan, Kansas 66506 (in prep.).


DATE
ILMED
8